On the Impact of Redirection on HTTP Adaptive Streaming Services in Federated CDNs

Jeroen Famaey[1], Steven Latré[1], Ray van Brandenburg[2], M. Oskar van Deventer[2], and Filip De Turck[1]

[1] Ghent University – iMinds, Belgium
[2] TNO, The Netherlands

Abstract. HTTP Adaptive Streaming (HAS) refers to a set of novel streaming services that allow clients to adapt video quality based on current network conditions. Their use of existing HTTP delivery infrastructure makes them perfectly suited for deployment on existing Content Delivery Networks (CDNs). Nevertheless, this leads to some new challenges, related to the distribution of content across servers and the latency caused by request redirection. The federation or interconnection of CDNs proliferates these problems, as it allows content to be distributed across networks and increases the number of redirects. This paper focuses on the second problem, assessing the impact of redirection on the Quality of Experience of HAS in CDN interconnection scenarios. Additionally, several novel inter-CDN request routing policies are proposed that aim to reduce the number of redirects. Our results indicate that redirection latency significantly impacts performance of HAS and more intelligent routing mechanisms are capable of solving this problem.

Keywords: HTTP Adaptive Streaming, Dynamic Adaptive Streaming over HTTP, Content Delivery Network interconnection, Quality of Experience

1 Introduction

The Content Delivery Network (CDN) market has traditionally been dominated by a small group of large players (e.g., Akamai, EdgeCast, and Limelight), which makes it difficult for smaller CDN providers to enter the field. This problem, combined with the recent rise of telco CDNs, has fuelled the idea of interconnecting or federating independent CDNs [1]. Such a federation has several advantages for all players involved, such as an increased geographical reach, content offloading during unexpected traffic spikes and the expansion of capabilities or functionalities [2].

Additionally, traditional multimedia streaming protocols are being superseded by HTTP-based adaptive streaming (HAS) services, such as Microsoft Smooth Streaming, Apple HTTP Live Streaming [3], Adobe HTTP Dynamic Streaming and MPEG Dynamic Adaptive Streaming over HTTP (DASH) [4]. HAS is an umbrella term for a wide range of HTTP-based streaming solutions

that allow client applications to dynamically adapt the quality of multimedia streams, in response to changes in network conditions [5,6]. To facilitate this, the content is split into temporal segments (usually between 2 and 10 seconds in duration), which are available in multiple quality representations. The segments, quality representations and their locations are listed in a manifest file, also called Media Description Presentation (MDP) in MPEG-DASH, which allows the client to retrieve the necessary segments. This novel approach has several important advantages, such as reliable transmission over TCP, reuse of existing HTTP server and caching infrastructure and compatibility with existing firewall and NAT configurations. These advantages make HAS services perfectly suited for delivery over CDNs, allowing the reuse of their existing HTTP-based delivery infrastructures [7,8]. The delivery of HAS-based services over CDNs introduces some novel opportunities, challenges and problems. First, the same content can be replicated on multiple server substrates or, due to the segmented nature of HAS content, can even be distributed across them. However, existing HAS client control algorithms do not support multiple alternative locations for a single segment and are optimized for streaming from a single server [9]. Second, CDNs employ the concept of request routing nodes, which serve as the client's access point to the network. Based on the request's parameters and context, request routers forward them to a suitable content server. This redirection process, usually performed using HTTP- or DNS-based redirects, can lead to significant latency, which is known to impact the Quality of Experience (QoE) of the HAS services [10].

As HAS services are an important part of current and future CDNs, their specific requirements and characteristics are also relevant in the design of CDN interconnection interfaces and protocols [11]. In fact, interconnection proliferates the problems that arise when delivering HAS services over CDNs even further. Specifically, segments of a single content item can now be distributed not only across servers within a single network, but even across multiple independently managed networks. Additionally, the number of redirects and the effects on QoE thereof only increase, as requests may now be redirected between CDNs as well.

In this paper, we focus on the second problem and evaluate the impact of HTTP-based redirection in federated CDNs on the QoE of HAS services. Although the effects on service quality of request routing have been previously studied [12], such studies only considered single CDN scenarios and did not investigate the problems associated with CDN interconnection and the more elaborate request routing policies required to handle them. Concretely, two novel request routing policies for federated CDNs are proposed. They employ dynamic manifest file rewriting [13] in order to significantly reduce the occurrence of redirects. Subsequently, the proposed novel policies are evaluated and compared to the traditional redirection-based policy using extensive simulation results.

The remainder of the paper is structured as follows. Section 2 introduces the three evaluated request routing policies and discusses their implementation complexity. The evaluated CDN-interconnection scenario, input parameters and evaluation metrics are discussed in Section 3. Subsequently, Section 4 presents

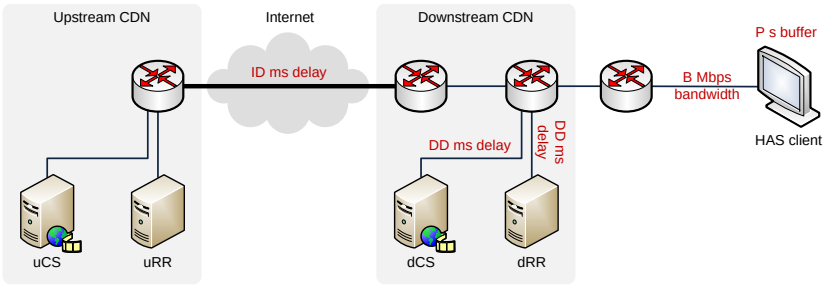


Fig. 1: An overview of network components involved in the delivery of HAS content over interconnected CDNs; a set of important parameters that affect QoE are denoted in red

the simulation results and discusses their relevance. Finally, the paper is concluded in Section 5.

2 Inter-CDN Request Routing

The interconnection of CDNs opens the door for novel optimizations when it comes to the delivery of HAS services. Due to the segmented nature of HAS content, a single video can easily be distributed across multiple CDNs and/or servers. This would, for example, allow a global CDN to offload the most popular parts of a movie (e.g., specific quality representations or even parts of a single representation) to local telco CDNs. Traditionally, the HAS manifest file points the client to the origin CDN's (from now on called the upstream CDN) request router (uRR), which would then redirect the client to another CDN's (called the downstream CDN) request router (dRR) if necessary. Once the request arrives at the correct CDN, the respective request router redirects the client one last time to the content server (uCS or dCS) hosting the actual segment. However, it is believed that such redirects could negatively influence the QoE of HAS services. In this paper, we evaluate this impact by comparing this traditional request routing policy to two novel policies that employ manifest file rewriting to reduce the number of redirects. Specifically, the segment locations in the manifest file are overwritten by the upstream and/or downstream CDN in order to point directly to the correct request router or even content server. Figure 1 depicts the important network components involved in the delivery of HAS content over interconnected CDNs. The three evaluated inter-CDN request routing policies are:

– ***UpstreamRR***: The manifest file points to the uRR for every segment. If the segment is located within the upstream CDN's network, the uRR sends the client a HTTP redirect request to point it to the correct uCS. Otherwise, the uRR redirects the client to dRR, which in turn redirects it to the correct dCS. Concretely, this approach results in either 1 or 2 indirections.

- ***DirectRR***: The manifest file immediately points to the correct request router, which redirects the client to the correct content server. This policy thus allows the client to circumvent going via the upstream CDN's network if the segment is located downstream. As such, this policy always results in 1 indirection.
- ***DirectCS***: The manifest file immediately points to the correct content server, which allows the client to download segments without being redirected. Compared to the *DirectRR* policy, the indirection of first contacting the request router is avoided, resulting in no indirections.

The *UpstreamRR* policy can be seen as the traditional CDN-interconnection approach, where clients always contact the origin CDN and HTTP redirection is used to point them to interconnected CDNs when necessary. It does not require any manifest file rewriting. Additionally, the upstream CDN does not need any detailed information about segment locations, as it only needs to redirect clients to the downstream request router. The *DirectRR* and *DirectCS* policies are more complex to implement, as they require the upstream CDN to rewrite the original manifest file. Additionally, when using the *DirectCS* policy, the downstream CDN either needs to share detailed chunk location information with the upstream CDN or the interconnected CDNs need to collaborate in creating the manifest file.

3 Evaluation Methodology

In this section, we discuss the evaluation methodology used to compare the three redirection policies for interconnected CDNs described in Section 2. The scenario used as a basis for the experiments consists of two interconnected CDNs. The downstream CDN is located close to the end-user (e.g., a telco CDN), while the upstream CDN is positioned further away (e.g., in the core Internet). The upstream CDN is assumed to be the main storage facility of the original content. As such, it hosts the manifest file but can offload content chunks to one or more downstream CDNs. Figure 1 graphically depicts the scenario and lists the parameters that were varied in the course of the experiments. The upstream CDN request router, upstream CDN content server, downstream CDN request router and downstream CDN content server are depicted as uRR, uCS, dRR, and dCS, respectively. During the experiments, five parameters were varied: the one-way Internet delay ID, the one-way downstream CDN delay DD, the per-client bandwidth B, the HAS client buffer size P, and the HAS segment duration S. The bandwidth on all other network links was set to 100 Mbps, while the one-way network delay was set to 5 ms. The round trip time (RTT) between two nodes can be calculated as the sum of the one-way delays of the links on the path between them, multiplied by two. In the performed experiments, the client and dRR/dCS are separated by three links, resulting in the total RTT $2 \times (2 \times 5 + DD) = (20 + 2 \times DD)$ ms. The client and uRR/uCS are 5 links apart, resulting the RTT $2 \times (4 \times 5 + ID) = (40 + 2 \times ID)$ ms. Note that

the processing delay on the CDN surrogates is not taken into account, as it is assumed to be negligible compared to the network delay.

The experiments evaluate a scenario where a single client downloads a 200 second video clip (split into 200/S segments). The first half is hosted by the downstream CDN, while the second half is hosted by the upstream CDN. The constant bitrate (CBR) video is available in 3 quality representations, with bitrates 500kbps, 1Mbps, and 2Mbps respectively.

As the end-user Quality of Experience (QoE) depends on several factors, multiple evaluation metrics are used in the comparison:

- **Average played quality**: The played quality layer, averaged over all segments and specified in terms of bitrate (Mbps).
- **Total buffer starvation time**: The accumulated time during which the next segment is not available at the client by the time it should start playing.
- **Start-up delay**: The time between the initial HTTP request for the first segment, performed by the client, and the time when the segment actually starts playing.

All reported results were obtained using the NS-3 simulation environment[3] in combination with the Network Simulation Cradle[4] (NSC). The used HAS client rate adaptation algorithm is based on the first version of the client algorithm incorporated in Microsoft's Smooth Streaming (MSS) client. The source code of this algorithm can be retrieved from CodePlex[5].

4 Results & Discussion

This section lists and discusses experimental results on the average played video quality, total buffer starvation time and the start-up delay. First, the effects of several parameters on the QoE metrics are studied, both in a congested and an uncongested network. Second, the influence of segment duration is evaluated.

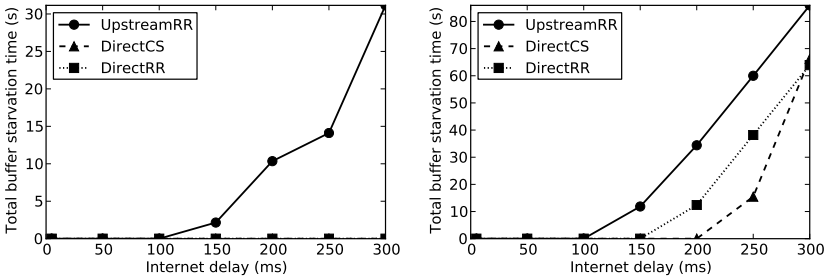
4.1 Congested Network Scenario

The congested scenario considers a client-side bandwidth B of 1Mbps, which, due to protocol overhead allows only the lowest 500kbps quality to be streamed. As such, this section focuses on a comparison of the buffer starvation time and start-up delay. The segment duration S is fixed at 2s. The results on buffer starvation as a function of one-way Internet delay ID with $DD = 5$ms, $B = 1$Mbps, $S = 2$s, and $P = 24$s are shown in Fig. 2. The starvation time is shown separately for the first 50 segments downloaded from dCS and the latter 50 downloaded from uCS. The results on start-up delay are depicted in Fig. 3 as a function of Internet delay ID and for different values of downstream CDN delay DD, with $B = 1$Mbps, $S = 2$s, and $P = 24$s.

[3] `http://www.nsnam.org/`

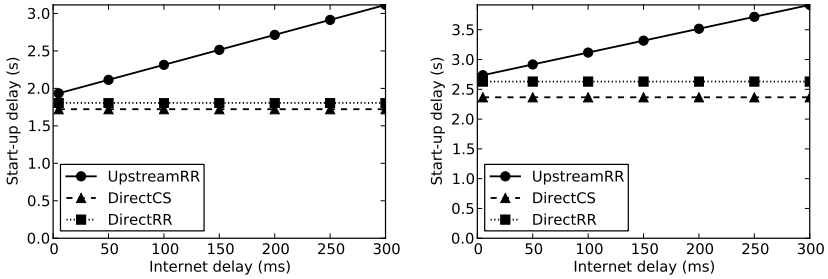
[4] `http://research.wand.net.nz/software/nsc.php`

[5] `https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming/`



(a) Downstream segments (b) Upstream segments

Fig. 2: The total buffer starvation time as a function of one-way Internet delay ID; for $DD = 5$ms, $B = 1$Mbps, $S = 2$s, and $P = 24$s



(a) $DD = 5$ms (b) $DD = 50$ms

Fig. 3: The start-up delay as a function of one-way Internet delay ID, for different values of one-way downstream CDN delay DD; for $B = 1$Mbps, $S = 2$s, and $P = 24$s

In general, the results depicted in Fig. 2 clearly show that minimising the number of HTTP redirects benefits the QoE significantly, both for segments hosted at the downstream as well as the upstream CDN. Specifically, several observations can be made based on the depicted results. First, as expected, *DirectCS* and *DirectRR* are not influenced by an increasing Internet delay ID for downstream segments, as they completely circumvent the upstream CDN in this case. In contrast, when using the traditional *UpstreamRR* approach buffer starvations start occurring at a one-way Internet delay of as low as 150ms. Second, for upstream segments, *DirectRR* and *DirectCS* are also negatively impacted by an increasing ID. However, *DirectCS* is significantly less influenced by it than *DirectRR*, as it circumvents *DirectRR*'s redirect from uRR to uCS.

The results in Fig. 3 confirm that the start-up delay is linearly proportional to the total RTT between the client and server. This explains the two evolutions

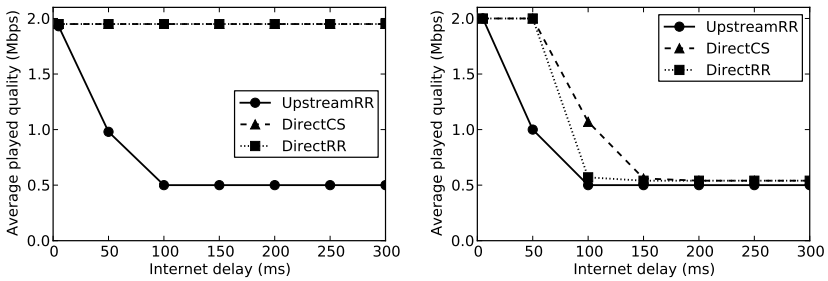
visible in the graphs. First, for segments hosted at the downstream CDN, the start-up delay for *UpstreamRR* increases as a function of the one-way Internet delay ID, while *DirectRR* and *DirectCS* are unaffected. Second, the start-up delay for all routing policies increases as a function of the one-way downstream delay DD. Finally, due to the lower redirection delay of *DirectCS* compared to *DirectRR*, the *DirectCS* start-up delay is slightly lower. Note that such start-up delay occurs whenever the buffer needs to be flushed. As such, this not only happens when a client initiates a session, but also for example when switching channels in Internet TV scenarios or skipping to another part of a video in a Video on Demand scenario.

In summary, it was shown that in congested scenarios, using the *DirectRR* or *DirectCS* routing policies can significantly reduce the amount of client-side buffer starvation compared to using the traditional *UpstreamRR* policy, when downloading HAS segments from the downstream CDN. Additionally, the use of *DirectCS* is beneficial in terms of buffer starvations compared to *DirectRR* and *UpstreamRR* when downloading segments from the upstream CDN. Finally, it was shown that the start-up delay is linearly proportional to the total RTT, both caused by network latency to the content server, as well as redirection delay. As such, the *DirectRR* and *DirectCS* start-up delay is unaffected by the Internet delay when streaming from the downstream CDN, while that of *UpstreamRR* is not. Moreover, *DirectCS* has a lower start-up delay than *DirectRR*.

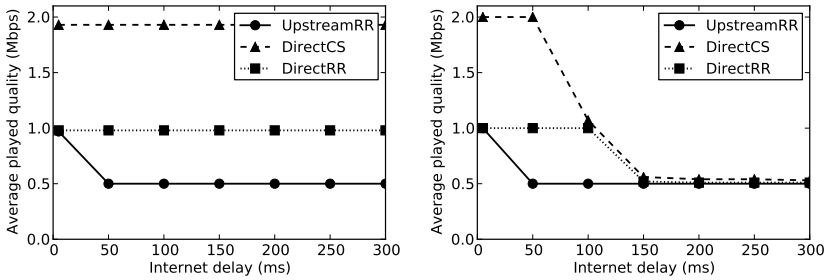
4.2 Uncongested Network Scenario

The uncongested scenario considers a client-side bandwidth B of 10Mbps, which is sufficient to download the highest 2Mbps quality layer stream. As such, this section does consider the delivered average video quality. As buffer starvation and start-up delay results were already discussed in much detail in the previous section, and they show similar trends in the uncongested scenario, they are omitted here. The segment duration S is once again fixed at 2s. The results on average played video quality as a function of one-way Internet delay ID, for different values of one-way downstream CDN delay DD and client buffer size P are shown in Figs. 4 and 5. The quality is shown separately for the first 50 segments downloaded from dCS and the latter 50 downloaded from uCS.

The results in Fig. 4 show that an increased number of redirections significantly impacts video quality, even for a relatively low RTT. Specifically, the results show that *UpstreamRR* achieves a significantly lower quality than *DirectRR* and *DirectCS* for segments hosted at the downstream CDN for all depicted parameter combinations. Additionally, as expected, the delivered video quality when using *UpstreamRR* is inversely proportional to the Internet delay ID. In contrast, *DirectRR* and *DirectCS* are unaffected. On the other hand, the downstream CDN delay DD does influence the quality of *DirectRR*. As depicted in Fig. 4c the quality of *DirectRR* and *UpstreamRR* suffers significantly from an increase in DD, while that of *DirectCS* is much less affected. In addition to the quality difference for segments hosted at the downstream CDN, there are also some remarkable differences for upstream CDN segments. Although *UpstreamRR*



(a) Downstream segments, $DD = 5$ms (b) Upstream segments, $DD = 5$ms

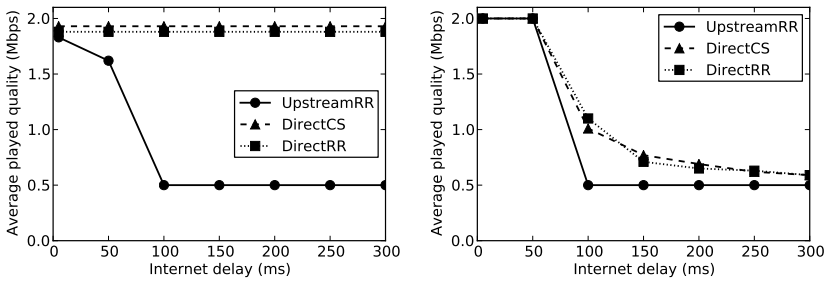


(c) Downstream segments, $DD = 25$ms (d) Upstream segments, $DD = 25$ms

Fig. 4: The average played quality as a function of one-way Internet delay ID, for different values of one-way downstream CDN delay DD; for $B = 10$Mbps, $S = 2$s, and $P = 6$s

and *DirectRR* exhibit the same behaviour when downloading segments from the upstream CDN, they do show a difference in video quality. This is due to the fact that *DirectRR* is able to fill up its buffer during the downstream streaming stage, while *UpstreamRR* is not. As the AVC MSS algorithm uses the buffer filling level in its quality adaptation decision, this results in a higher quality for *DirectRR*. Finally, the results in Fig. 5 show that increasing the buffer size leads to a higher delivered video quality in almost all cases.

In summary, the merits of *DirectRR* and *DirectCS* compared to *UpstreamRR* in uncongested networks were clearly shown. In addition to a reduction in buffer starvations and start-up delay, the *DirectRR* and *DirectCS* policies also result in an increased average video quality. Specifically, when using *UpstreamRR* and streaming content from the downstream CDN, the video quality is significantly impaired by an increase in Internet delay, while *DirectRR* and *DirectCS* are unaffected. Additionally, a high downstream CDN delay DD significantly reduces the video quality when using *DirectRR* or *UpstreamRR*, while it has much less impact on *DirectCS*.



(a) Downstream segments (b) Upstream segments

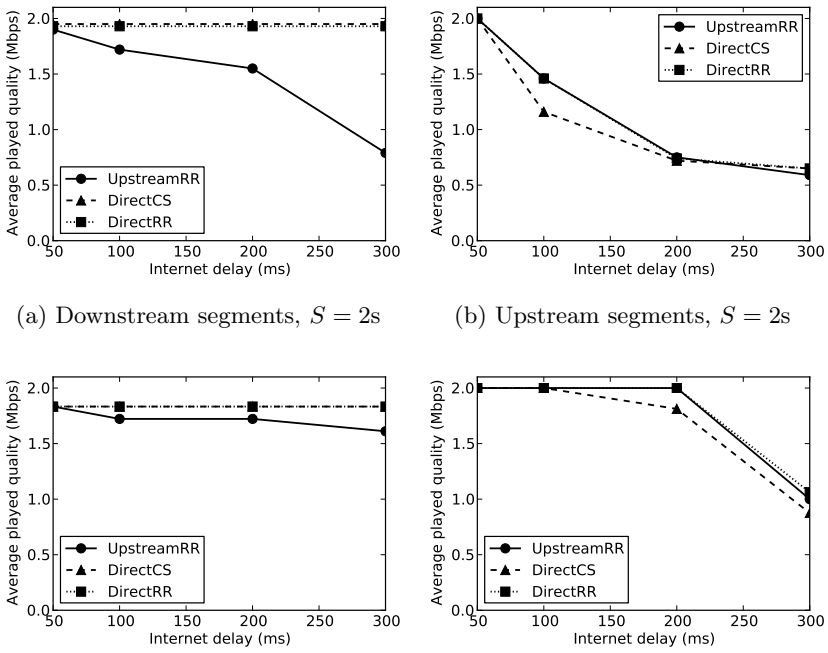
Fig. 5: The average played quality as a function of one-way Internet delay ID; for $DD = 25$ms, $B = 10$Mbps, $S = 2$s, and $P = 24$s

4.3 Influence of Segment Duration

The previously presented results only considered a short segment duration S of 2s. Although this value is widely used, by for example MSS-based services, others, such as Apple HTTP Live Streaming (HLS), generally recommend longer segment durations. This section evaluates the effect of segment duration S on the average played quality as well as the start-up delay in an uncongested scenario with a client-side bandwidth B of 5Mbps. As results showed that the used HAS algorithm performs poorly if the buffer fits only two segments or less and a segment duration of up to 12s is considered, a buffer size P of 36s is used. The results on average played quality as a function of one-way Internet delay ID and for different segment durations S are shown in Fig. 6. The quality is shown separately for the first 50 segments downloaded from dCS and the latter 50 downloaded from uCS. The results on start-up delay are also depicted in Fig. 7 as a function of ID and for different S.

The results depicted in Fig. 6 clearly indicate that increasing the segment duration greatly improves performance of the three routing policies for large delays. If the delay is large enough, it evens out performance, in terms of video quality, of the three policies almost completely, significantly reducing the negative effects of redirects. This is obviously due to the fact that increasing the segment duration, decreases the number of requests and thus the relative delay introduced by redirects. However, longer segment durations also result in slightly lower average quality when the delay is very low (e.g., $ID = 50$ms, and $DD = 5$ms). This is because increasing the segment duration results in a proportional increase in convergence time to the optimal video quality.

Although using longer segment durations is an effective way to increase the video quality in face of redirects with high latency, it also has several disadvantages. As shown in Fig. 7, the start-up delay increases significantly as a function of the segment duration. This is the case for all routing policies. Additionally, long segment durations are usually a poor choice in combination with live ser-



(a) Downstream segments, $S = 2$s

(b) Upstream segments, $S = 2$s

(c) Downstream segments, $S = 12$s

(d) Upstream segments, $S = 12$s

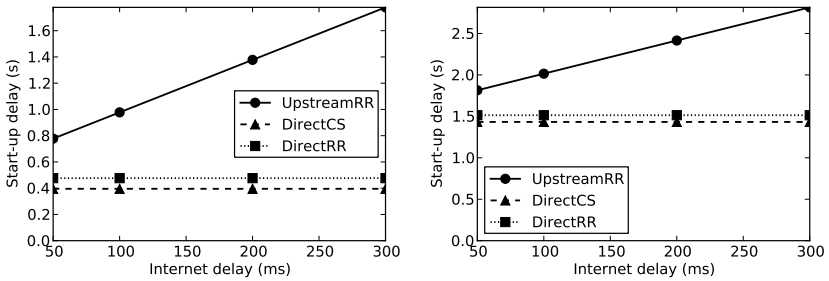
Fig. 6: The average played quality as a function of one-way Internet delay ID, for different values of segment duration S; for $DD = 5$ms, $B = 5$Mbps, and $P = 36$s

vices, as they lead to significant lag of the streaming session compared to the live time.

In summary, results showed that increasing the HAS segment duration can help in overcoming the reduced video quality that occurs when using simple Inter-CDN routing policies, such as *UpstreamRR*. Nevertheless, long segment durations also have significant disadvantages, such as slower convergence to the optimal quality, an increased start-up delay and greater session lag in live scenarios. This makes them unsuitable in many use cases, such as live, interactive or channel-switching intensive services.

5 Conclusion

In this paper, we proposed, evaluated and compared several policies for routing requests and retrieving HAS content segments distributed across multiple interconnected CDNs. Concretely, the traditional policy, herein called *UpstreamRR*, in which the original CDN's request router dynamically redirects the end-users



(a) $S = 2$s (b) $S = 12$s

Fig. 7: The start-up delay as a function of one-way Internet delay ID, for different values of segment duration S; for $DD = 5$ms, $B = 5$Mbps, and $P = 36$s

towards the CDN currently hosting the requested content, is compared to two novel policies, called *DirectRR* and *DirectCS*. These novel policies employ HAS manifest file rewriting to immediately point end-users to the correct CDN (*DirectRR*) or even the correct content server (*DirectCS*).

A thorough evaluation, using an open source implementation of the Microsoft Smooth Streaming client algorithm and based on NS-3 simulation results, was conducted. It shows that the QoE suffers greatly as a consequence of the HTTP redirects that occur when employing the standard *UpstreamRR* policy. Specifically, it was shown that when downloading segments from the downstream CDN, *DirectRR* and *DirectCS* result in a much lower buffer starvation rate and start-up delay, as well as an increased video quality compared to *UpstreamRR*. Additionally, *DirectCS* significantly outperforms the other two strategies in terms of buffer starvation rate and delivered video quality for segments downloaded from the upstream CDN. Finally, the evaluation showed that increasing the segment duration can negate the negative effects of redirects on video quality when using the *UpstreamRR* policy. However, it also leads to increased start-up delay, slower convergence to the optimal quality and most importantly a significant increase in lag of streaming sessions compared to the live time.

In summary, these results indicate the need for advanced request routing mechanisms, as well as extensive cooperation between interconnected CDNs, to be able to satisfy end-user quality requirements of state-of-the-art HAS-based services. Additionally, the results show the merits of the more complex *DirectCS* policy compared to the easier to implement *DirectRR*.

References

1. Peterson, L., Davie, B.: Framework for CDN interconnection. Internet-Draft draft-ietf-cdni-framework-02, IETF Secretariat (2012)

2. Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., Watson, G.: Use cases for content delivery network interconnection. Internet-Draft draft-ietf-cdni-use-cases-10, IETF Secretariat (2012)
3. Pantos, R., May, W.: Http live streaming. Internet-Draft draft-pantos-http-live-streaming-10, IETF Secretariat (2012)
4. Stockhammer, T.: Dynamic adaptive streaming over HTTP – standards and design principles. In: Second annual ACM conference on Multimedia systems. (2011) 133–144
5. Jarnikov, D., Özçelebi, T.: Client intelligence for adaptive streaming solutions. Signal Processing: Image Communication **26**(7) (2011) 378–389
6. Sanchez, Y., Schierl, T., Hellge, C., Wiegand, T., Hong, D., De Vleeschauwer, D., Van Leekwijck, W., Le Louedec, Y.: Efficient HTTP-based streaming using scalable video coding. Signal Processing: Image Communication **27**(4) (2012) 329–342
7. Cicco, L., Mascolo, S.: An experimental investigation of the akamai adaptive video streaming. In Leitner, G., Hitz, M., Holzinger, A., eds.: HCI in Work and Learning, Life and Leisure. Volume 6389 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 447–464
8. Liu, C., Bouazizi, I., Hannuksela, M.M., Gabbouj, M.: Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. Signal Processing: Image Communication **27**(4) (2012) 288–311
9. Pu, W., Zou, Z., Chen, C.W.: Dynamic adaptive streaming over HTTP from multiple content distribution servers. In: IEEE Global Telecommunications Conference (GLOBECOM). (2011) 1–5
10. Famaey, J., Latré, S., Bouten, N., Van de Meerssche, W., De Vleeschauwer, B., Van Leekwijck, W., De Turck, F.: On the merits of SVC-based HTTP adaptive streaming. In: 12th IFIP/IEEE International Symposium on Integrated Network Management (IM). (2013)
11. van Brandenburg, R., van Deventer, O., Le Faucheur, F., Leung, K.: Models for adaptive-streaming-aware CDN interconnection. Internet-Draft draft-brandenburg-cdni-has-04, IETF Secretariat (2013)
12. Masa, M., Parravicini, E.: Impact of request routing algorithms on the delivery performance of content delivery networks. In: IEEE International Performance, Computing, and Communications Conference. (2003) 5–12
13. Houdaille, R., Gouache, S.: Shaping HTTP adaptive streams for a better user experience. In: Third annual ACM conference on Multimedia systems. (2012) 1–9